Avant!

Chapter 28

Performing Library Encryption

Library encryption allows you to distribute proprietary Star-Hspice models, parameters and circuits to other people without revealing your company's sensitive information. Recipients of an encrypted library can run simulations that use your libraries, but Star-Hspice does not print encrypted parameters, encrypted circuit netlists, or internal node voltages. Your library user sees the devices and circuits as "black boxes," which provide terminal functions only.

Use the library encryption scheme primarily to distribute circuit blocks with embedded transistor models, such as ASIC library cells and I/O buffers. Star-Hspice uses subcircuit calls to read encrypted information. To distribute device libraries only, create a unique subcircuit file for each device.

This chapter describes Avant!'s Library Encryptor and how to use it to protect your intellectual property. The following topics are covered in the chapter:

Understanding Library Encryption

The library encryption algorithm is based on that of a five-rotor Enigma machine. The encryption process allows the user to specify which portions of subcircuits are encrypted. The libraries are encrypted using a key value that Star-Hspice reconstructs for decryption.

Controlling the Encryption Process

To control the beginning and end of the encryption process, insert .PROTECT and .UNPROTECT statements around text to be encrypted in an Star-Hspice subcircuit. The encryption process produces an ASCII text file in which all text that follows .PROTECT and precedes .UNPROTECT is encrypted.

Note: The Star-Hspice .PROTECT and .UNPROTECT statements often are abbreviated to .PROT and .UNPROT, respectively. Either form may be used in Star-Hspice input files.

Library Structure

The requirements for encrypted libraries of subcircuits are the same as the requirements for regular subcircuit libraries. Subcircuit library structure requirements are described in Chapter 2, “Getting Started.” Refer to an encrypted subcircuit by using its subcircuit name in a subcircuit element line of the Star-Hspice netlist.

The following example provides the description of an encrypted I/O buffer library subcircuit. This subcircuit is constructed of several subcircuits and model statements that you need to protect with encryption. Figure 28-1: shows the organization of subcircuits and models in libraries used in this example.

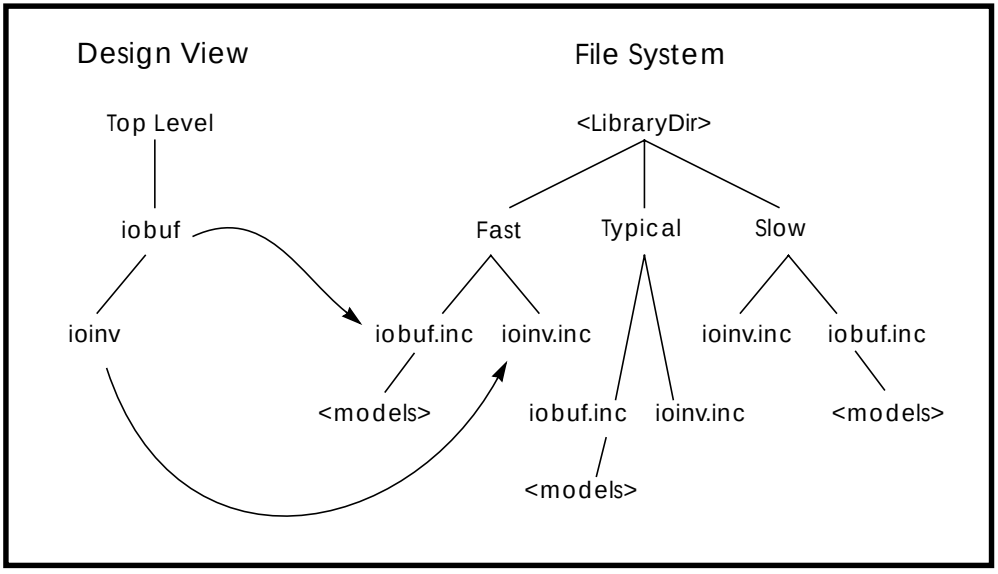

Figure 28-1: Encrypted Library Structure

The following input file fragment from the main circuit level selects the *Fast* library and creates two instances of the *iobuf* circuit.

```
...
.Option Search='<LibraryDir>/Fast'          $ Corner Spec
x1 drvin drvout iobuf Cload=2pF             $ Driver
u1 drvout 0 recvin 0 PCBModel...            $ Trace
x2 recvin recvout iobuf                     $ Receiver
...
```

The file *<LibraryDir>/Fast/iobuf.inc* contains:

```
.Subckt iobuf Pin1 Pin2 Cload=1pF
*
* iobuf.inc - model 2001 improved iobuf
*
.PROTECT
cPin1 Pin1 0 1pF           $ Users can't change this!
x1 Pin1 Pin2 ioinv         $ Italics here means encrypted

.Model pMod pmos Level=28 Vto=...      $ <FastModels>
.Model nMod nmos Level=28 Vto=...      $ <FastModels>
.UNPROTECT
cPin2 Pin2 0 Cload         $ give you some control
.Ends
```

The file *<LibraryDir>/Fast/ioinv.inc* contains:

```
.Subckt ioinv Pin1 Pin2
.PROTECT
mp1 Vcc Pin1 Pin2 Vcc pMod...          $ Italics=Protected
mn1 Pin2 Pin1 Gnd Gnd nMod...          $ Italics=Protected
.UNPROTECT
.Ends
```

After encryption, the basic layout of the subcircuits is the same. However, the text between .PROTECT and .UNPROTECT statements is unreadable, except by Star-Hspice.

The protection statements also suppress printouts of encrypted model information from Star-Hspice. Only Star-Hspice knows how to decrypt the model.

Knowing the Encryption Guidelines

In general, there are no differences between using the encrypted models and using regular models. However, you *must test your subcircuits before encryption*. You will not be able to see what has gone wrong after encryption because of the protection offered by Star-Hspice.

Use any legal Star-Hspice statement inside your subcircuits to be encrypted. Refer to ".SUBCKT or .MACRO Statement" on page 22-9 for further information on subcircuit construction. You must take care when structuring your libraries. If your library scheme requires that you change the name of a subcircuit, you must encrypt that circuit again.

Placement of the .PROTECT and .UNPROTECT statements allows your customers to see portions of your subcircuits. If you protect only device model statements in your subcircuits, your users can set device sizes or substitute different subcircuits for lead frames, protection circuits, and so on. This requires your users to know the circuits, but it reduces the library management overhead for everyone.

Note: If you are running any version of the encryptor prior to Star-Hspice Release H93A.03, there is a bug that prevents Star-Hspice from correctly decrypting a subcircuit if that subcircuit contains any semicolon (;) characters, even in comments.

In the following example, the subcircuit *badsemi.dat* is encrypted into *badsemi.inc*.

```
* Sample semicolon bug
.SubCkt BadSemi A B
.PROT
* Semicolons (;) cause problems!
r1 A B 1k
.UNPROT
.Ends
```

Star-Hspice responds with the following message:

```
 **reading include file=badsemi.inc
 **error**:  .ends  card missing at readin
 >error   ***difficulty in reading input
```

To solve this problem, remove the semicolon from *badsemi.dat* and encrypt the file again.

Some versions of Star-Hspice cannot decrypt files with lines longer than 80 characters. Avant! strongly recommends that *all* encrypted files be limited to an 80-character line length, because at encryption time the Star-Hspice version that the customer uses is unknown.

You cannot gather the individually-encrypted files into a single file or include them directly in the Star-Hspice netlist. Place them in a separate directory pointed to by the .OPTION SEARCH = <*dir*> named <*sub*>*.inc* for correct decryption by Star-Hspice.

Installing and Running the Encryptor

This section describes how to install and run the Encryptor.

Installing the Encryptor

If Star-Hspice is already installed on your system, place the Encryptor in the directory *$installdir/bin* to install it. Add the lines that allow the Encryptor to operate to your *permit.hsp* file in the *$installdir/bin* directory.

If Star-Hspice is not installed on your system, first install Star-Hspice according to the installation guide and *Star-Hspice Release Notes* included in your Star-Hspice package, and then follow the instructions in the previous paragraph.

Note: If you are running a floating license server, you must stop and restart the server to see the changes to the permit file.

Running the Encryptor

The Encryptor requires three parameters for each subcircuit encrypted: *<InFileName>*, *<OutFileName>*, and the key type specifier, *Freelib*. Enter the following line to encrypt a file.

```
metaencrypt -i <InfileName> -o <OutFileName> -t Freelib
```

As the Encryptor reads the input file, it looks for .PROT/.UNPROT pairs and encrypts the text between them. You can encrypt only one file at a time.

To encrypt many files in a directory, use the following shell script to encrypt the files as a group. This script produces a *.inc* encrypted file for each *.dat* file in the current directory. The procedure assumes that the unencrypted files are suffixed with *.dat*.

```
#!/bin/sh
for i in *.dat
   do
         Base=‘basename $i .dat‘
         metaencrypt -i $Base.dat -o $Base.inc -t Freelib
   done
```

```
.SUBCKT ioinv Pin1 Pin2
.PROT FREELIB              $ Encryption starts here ...
X34%43*27@#^3rx*34&%^#1
^(*^!^HJHD(*@H$!:&*$
dFE2341&*&)(@@3            $ ... and stops here (.UNPROT is
encrypted!)
.ENDS
```